Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 3 (2011) 1512–1516

Procedia Computer Science

www.elsevier.com/locate/procedia

WCIT 2010

Using Neural Search Approach for Resource Discovery in P2P Networks

Hesam Yousefipour*, Zahra Jafari

Department of Information Technology, Institute for Advanced Studies in Basic Sciences, P.O. Box 45195-1159, Zanjan 45195, Iran

Abstract

Resource discovery is one of the main concerns of Peer-to-Peer (P2P) networks. This is due to the distributed nature of P2P networks and that there is no centralized indexing to look for the information of available resources and their places. Previously, to search for one requested resource, algorithms were invented that exploited the local knowledge about the network. This approach is inaccurate and cannot be scaled well in presence of large P2P networks. On the other hand, replicating each node's local knowledge and using it to build up a comprehensive knowledge base is very expensive. In this paper, we describe a method that benefits from the flexibility of Neural Networks and not only takes into account most of the measures previously used to optimize the searching process, but also performs efficiently and is designed simple. Result of running the algorithm on a sample network is evaluated and its performance is compared with another common method to prove its efficiency.

Selection and/or peer-review under responsibility of the Guest Editor.

Keywords: Peer to Peer Network; Resource Discovery; Multi Layer Perceptron; Neural Network

1. Introduction

It is an integral quality of the nowadays Internet that a dramatic amount of resources exist for all kinds of services. Many network structures have been developed to benefit from this fact, and to overcome the challenges posed as a result of such environment. P2P networks are one such example. As peers opt to interact with one another to find needed resources, the means of interaction between peers become important. Understanding this importance is the key to find methods that help systems inter-operate smoothly, without raising the need for users to learn every detail of the system and of the peers they share their network with.

Formerly, P2P networks duplicated resource indexes to be able to benefit from the convenience of local searching, but this method required a huge traffic that could affect the system performance severely. Centralized discovery mechanisms also did not scale well in large P2P networks that are distributed in nature and suffer from a single point of failure. Modern networks use decentralized searching as an alternative and conduct directed search requests. Decentralized searching was first done by flooding search requests out across peers that could pose a risk to the network availability in high traffic volumes. More efficient directed search strategies, including super nodes

* Hesam Yousefipour. Tel.: +98-912-296-4404; Fax: +98-241-415-2182
E-mail address: syousefi@iasbs.ac.ir

1877-0509
doi:10.1016/j.procs.2011.01.040

and distributed hash tables, are now used to address the flexibility issues [1, 2].

In Breadth First Search (BFS) [3], a requester node sends a query to all its neighbors, causing them to do the same with the exception that they do not pass the query to the node they received it from. Each node is instructed to drop a query if it decides that the received query is already handled, and to control the query flow, Time-to-Live (TTL) mechanism is adopted. One advantage of using BFS is that it guarantees to find a requested resource if there exist one; provided that TTL be picked high enough. The downside of the algorithm however is that, it is not scalable for large networks due to the high traffic it incurs.

Neural Networks can be used to improve the searching mechanism efficiency by choosing appropriate hops for query path intelligently. Whenever a query reaches its destination node, the node replies back to sender from the same path the query has traversed. An efficient approach is one that in the minimum time could locate as much resources with a given ID as possible while causing the minimum traffic. In this paper, we propose an algorithm using Neural Networks that is more efficient. We then evaluate it against scalability criteria and compare it to BFS.

In section 2, references to previous works done in P2P resource discovery are given. Section 3 describes the Neural Search Approach. Section 4 deals with the simulation environment and test case. Evaluation and analysis of the results are studied in the Section 5. Finally, in section 6, we conclude the efficiency of the algorithm using the experimental results.

2. Related Work

P2P Networks in essence are categorized by three main architectures: Centralized, Decentralized but Structured and Decentralized and not Structured. Centralized Networks mainly rely on some shared information available through a specific node; examples of such networks include Napster, which uses a constantly updated directory residing on central locations. Some other P2P networks benefit from a decentralized but tightly controlled structure, which is the case for Freenet Network [4]. Another scenario is when a Network is decentralized and at the same time loosely structured which is the case for most of the current active P2P Networks where the network is formed dynamically by ad-hoc additions of nodes. Our focus is on resolving such generic resource discovery challenges.

Many of researches concerning the resource discovery problem centre their attention on scale-free networks and utilize the topological properties of the network whose degree distribution follow a power-law relationship [5]. They mostly start the searching algorithm by proceeding toward highest-degree nodes to exploit the possible links between such nodes and the entire network nodes.

Considering different searching and replication strategies, [3] and [6] have evaluated BFS and concluded that it is not scalable especially on power-law network graphs where flooding decreases the performance disastrously. Increasing TTL value gradually is one of the early solutions suggested for the problem, but the algorithm still suffered from lack of scalability. Taking successive steps, each in a random direction by several walkers was another remedy to the problem called Random Walkers.

Some other methods, consider doing minor alternations to BFS algorithm in order to compensate for the BFS flooding negative effects [7]. Examples of such ideas include Modified Random BFS, where the researchers designed an algorithm that selects a random subset of neighbors to forward the query.

All previous methods are only applicable to specific circumstances where some fixed factors determine the functionality of the algorithm. Another limitation with the proposed techniques is that their controlling factors could not be adapted dynamically and extensive reliance on manual configuration especially in dynamic environments like P2P networks may be practically impossible.

Neural Networks have the competitive advantage of considering the full potential of the environment without raising the need of manual configuration and being able to adapt autonomously to the environment. However due to a complete distributed nature, they may present an unpredictable and complex system specially when mixed with evolutionary approaches. Another challenge in using Neural Networks is that they may prove to be very slow if poorly designed and could not become handy in network traffic bursts or real time situations.

In an effort to design an algorithm that is more adaptable to the dynamic nature of P2P networks, [8] has proposed an intelligent approach to alleviate the flexibility issues of previous algorithms using neural networks and Evolutionary algorithms. However, there seems to be some issues in this study that should be resolved:

- Simulation time in large networks is long
- Building complex combination of multiple inputs is quite risky if the behavior of individual inputs is not known

and the result might not be easily explained.

To address the discussed matters, Neural Search Approach takes into account only a few essential measures while keeping the network graph as simple as possible.

3. Neural Search Approach

We propose an algorithm called Neural Search Approach (NSA), which tries to decide to whom of the neighbors the received request should be forwarded. This is done based on the output of a multilayer perceptron neural network. For the required scalability and justifiability of the result, the algorithm uses the simplest possible architecture and is run for every of the current node neighbors. NSA can be represented as follows:

$$O: \boldsymbol{I} \rightarrow \{0,1\}, \boldsymbol{I} \in [0,1]^3. \quad (1)$$

The Input is a 3-dimentional vector denoting the state of the received query. Activation function in the output layer is the threshold function, therefore the output of the network would be whether one, in case of choosing to forward the query to the specified neighbor, or zero, in case of choosing not to forward the query to the specified neighbor. The neural network is used in each node for each neighbor to decide which could deliver the query to its destination. The inputs are chosen as follows:

- *Bias*, is the network bias and has the value of one.
- *Hops*, is the number of hops the query has traversed so far.
- *Neighbors*, is the number of the neighbors of the recipient node.

Hops is scaled using the function *f(x) = 1/x+1* and *Neighbors* is scaled using the function *f(x) = 1/x*. This is to ensure that all the inputs are between zero and one. There are two hidden layers in the network. The first one has 10 and the second one has 5 hidden nodes. From experience, we use Tanh as the activation function for hidden nodes. Activation functions are as illustrated in Equation 2 and 3.

$$t(a) = \frac{2}{1+e^{-2a}} - 1, \quad \text{for Hidden Layers.} \quad (2)$$

$$s(a) = \begin{cases} 0, & if\ a < 0 \\ 1, & if\ a \geq 0 \end{cases}, \quad \text{for Output Layer.} \quad (3)$$

All together, the output of the neural network can be calculated with Equation 4. I_i denotes the input parameter and w_{xy} denotes the neural network weights on layer x in position y.

$$O = s(1 + \sum_{k=1}^{5} w_{2k} t(\sum_{k=1}^{10} w_{1k} f(I_i))). \quad (4)$$

3.1. Perceptron Learning

The only unknown values in the algorithm remain the weights that should be initialized and trained in order for the algorithm to function. To find the optimum weights, we need some training sets. We study a sample P2P network for training and then test the result using a separate generalization set from the same P2P network.

The sample P2P network was generated using the Barabasi-Albert model [9] in MATLAB, so the network's node distribution followed the power-law characteristics. Power-law networks have the quality of hosting many nodes with low degrees, which is the case for most of the P2P networks in use including the Internet.

Training set is better to be consisted of carefully selected data to represent the properties of the entire network. After training, we can test the network to determine its ability to generalize. While testing the trained network, to avoid over-fitting we may stop the training if the performance of the system started to decrease. Generalization set is used to test the trained network against complete random data to determine if the network is functioning as planned.

For training purposes, we use Floyd-Warshall algorithm to determine shortest paths available from the query originator to all the nodes that host the requested resource. This data is saved in a matrix and then used to define the

target set. The selected query is checked for the resource it is seeking and its neighbors are evaluated to decide if each could lead to a shortest path to one of the requested resource.

To train the Neural Network, we use the Back-propagation algorithm. Training is done a few times for each of the nodes that are being decided upon which of their neighbors to send the query. Resource instances were chosen to scatter throughout the network. Each of the nodes could acquire as many types of resources as its links to its adjacent neighbors. This guarantees that a node with the highest degree had only one instance of each of the resources and nodes with less links acquired even less resources picked randomly from the resource pool.

4. Implementation

To avoid excessive processing load, we chose a network with 100 nodes whose degree distribution followed the Power-law. To generate the Network, a seeding matrix of size 2x2 was used and each newly generated node was restricted to make link only to one of the other peers in the network. This makes sure that there would not exist many ways for each query to travel from its current node toward its destination.

To obtain a rational presentation of the algorithm efficiency, we thought it enough to generate 200 queries, each randomly placed in the network and assigned with one of the five resources available.

Network matrix along with vectors denoting hosted resources and queries were passed to the Neural Search Approach function for training. This function processed through a loop to train the Neural Network for each of the queries in every node. After the training was done with the node currently hosting the query, query states are updated as necessary and control is passed to the for loop for another node processing. Each query generates one to several query states and the Neural Network was trained 10 epochs for each query. This process continued in a loop and the Network gradually adapted itself to all the queries selected for training. Fig. 1 illustrates the state of the Neural Network after all training set queries were used to train the network. As shown in this figure, the error of the generalization data has gradually decreased to less than 10^{-30} after the 7^{th} epoch.

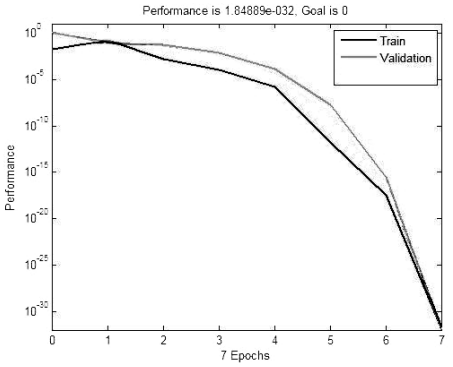

Fig. 1. State of the Neural Network in the last step of training. The darker curve indicates the data set used to train the data. Both validation set and train set error has decreased throughout the training process

5. Simulation Results

To evaluate the performance difference between BFS and Neural Search Approach, we let the two algorithms run until all the queries were eventually reached one of their destinations. Throughout the time required for such process, some of the queries managed to find all the instances of the queried resources and some less.

To achieve a better conclusion, we compared NSA with both a BFS with TTL 2 and TTL 3. The result is shown in Table 1. Hops is the number of overall hops that is traversed in order for the algorithm to reach its goal. Resources indicate the number of overall resources each algorithm could find as long as all the queries could find at least one resource of their choice. Efficiency is measured by calculating the ratio between the located resources and used query packets.

Table 1. Efficiency of the algorithms

Algorithm	Hops	Resources	Efficiency	Performance
BFS2	122	83	0.2132	1.4698
BFS3	247	188	0.1642	1.3138
NSA	98	132	0.2978	0.7424

When there are only a few resources to locate, efficiency is easier to keep high because usually those can be reached from the central nodes of the network. However, when the number of resources in the network increases, query has to move across the edges of the network to reach as many resources as needed.

Performance is the result of dividing Hops by Resources. On average, BFS2 takes 1.46 hops to find one resource and BFS3 takes 1.31 hops. Both of the values are greater than one while the NSA takes 0.74 hops to find one resource on average. Table one results can prove the efficiency of the proposed algorithm.

6. Conclusion

We proposed a new resource discovery algorithm that we claimed to be faster in finding the available resources in P2P network while at the same time efficient in terms of the packets generated for searching purposes.

By analyzing the behavior of Neural Search Approach and BFS searching algorithm, we came to the conclusion that NSA can find the resources within a P2P network approximately two times faster than BFS algorithm. Neural Search Approach seems to prefer central nodes early in the query and uses multiple paths for doing this. NSA also takes into account the Power-Law characteristic of real P2P networks. NSA finds the resources according to all the features of the environment and adapts itself to the probable changes in the network without having to take into account many measures used to find the minimum-length paths by previous algorithms.

While NSA performs well compared to BFS, it is yet to be studied more in depth. One reason for such claim is that, it still does not exploit history based searching even though they can maximize the efficiency of the algorithm significantly. Another yet to be answered question about the NSA is that we are not yet very certain about the changes in the internal structure of the Neural Network used for training. One thing that is not included in this study is scalability factors or finding what happens if the P2P network grows too huge. In addition, for how long a well designed Neural Network can be efficient in case of dynamic network changes. Yet another issue that is still a concern is the speed of training. While we chose a structure for the Neural Network that is rather simple and yet performs well, we aim at finding ways of increasing training speed in the presence of very large networks without causing performance to suffer.

References

1. Y. Chawathe, S. Ratnasamy and L. Breslau, Making Gnutella-like P2P Systems Scalable, SIGCOMM, 2003.

2. M. F. Kaashoek and D. R. Karger, Koorde: A simple degree-optimal hash table, in Proceedings of IPTPS, Berkeley, CA, Feb 2003, pp. 98-107.

3. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, Search and Replication in Unstructured Peer-to-Peer Networks, in Proceedings of the 16th International Conference on Supercomputing, ACM Press, 2002, pp. 84-95.

4. Open Source Community, The free network project - rewiring the internet, In http://freenet.sourceforge.net/, 2001.

5. B. J. Kim, C. N. Yoon, S. K. Han and H. Jeong, Path finding strategies in scale-free networks, Physical Review E 65, 2002.

6. A. Barabsi, Linked, Perseus Publishing, 2002.

7. V. Kalogeraki, D. Gunopulos and D. Zeinalipour-Yatzi, A Local Search Mechanism for Peer-to-Peer Networks, in Proceedings of the 11th International Conference on Information and Knowledge Management, ACM Press, 2002, pp. 300-307.

8. Vapa, M., Kotilainen, N., Auvinen A., Kainulainen, H. and Vuori, J., Resource discovery in p2p networks using evolutionary neural networks, Intern. Conf. on Advances in Intelligent Systems - Theory and Applications, 067-04, 2004.

9. A. L. Barabsi and R. Albert, Emergence of Scaling in Random Networks, Science 286, pp. 509-512, 1999.